Adabas®

Installation Manual (BS2000)

Manual Order Number: ADA742-010BS2

This document applies to the Adabas software package at version 7.4 and to all subsequent versions, unless otherwise indicated in new editions or technical newsletters.

Specifications contained herein are subject to change, and these changes will be reported in subsequent revisions or editions.

Readers' comments are welcomed. Comments may be addressed to the Documentation Department at the address on the back cover or to the following e-mail address:

Documentation@softwareag.com

TABLE OF CONTENTS

ABOUT THIS MANUAL

This manual provides information for installing and configuring Adabas version 7.4 on Siemens BS2000 systems.

See page 5 for the specific operating system levels supported by this Adabas release.

Operating system requirements are provided, as well as procedures for installing Adabas, for connecting Adabas to TP monitor subsystems such as TIAM or UTM, and for adding new I/O devices.

Who Should Read and Use This Manual

This manual is intended for those who plan or perform Adabas installation, and for those who manage or maintain an Adabas database system (such as database administrators and systems programming personnel).

How the Manual is Organized

Chap	Content
1	provides pre-installation information: supported environments, a way to check your Adabas version using a program.
2	provides the job control statements, step-by-step procedures, and system-related features and considerations for installing Adabas under BS2000.
3	tells you how to install Adabas with TIAM, UTM, and other teleprocessing monitors supported by Adabas.
4	describes device types, provides special device type considerations and information about using ZAPs to include new devices in the system configuration.
5	tells you how to install the demo version of Adabas Online System (AOS) included with Adabas.
6	tells you how to install the Adabas Recovery Aid.
7	describes the dump formatter ADAFDP.
A	describes the BS2000-compatible ADAUTM component, which supports Adabas operation with the SNI Universal Transaction Monitor™.
B	reproduces the delivered EBCDIC/ASCII translation tables.
C	is a glossary of Adabas terms.

Other Manuals You May Need

The following Software AG manuals are referred to in this manual; they may be needed when installing Adabas:

- Adabas Release Notes (ADAvrs-008IBB)
- Adabas Operations Manual (ADAvrs-110IBB);
- Adabas DBA Reference Manual (ADAvrs-030IBB);
- Adabas Triggers and Stored Procedures Manual (ADAvrs-032IBM)
- Adabas Dynamic Caching Manual (ADAvrs-034IBB)
- Adabas Command Reference Manual (ADAvrs-050IBB);
- Adabas Messages and Codes (ADAvrs-060IBB);
- Adabas Utilities Manual (ADAvrs-080/81IBB—two volumes).
- Adabas Security Manual (This manual is available only at the written request of an authorized user site representative);
- Adabas Online System Manual (AOSvrs-030IBB);
- Adabas Delta Save Facility Manual (ADEvrs-030IBB);
- Adabas Parallel Services Manual (ASM742-030IBB)

For Software AG's System Maintenance Aid (SMA) information, see the manual *System Maintenance Aid Manual* (SMAvrs-030IBB).

The "vrs" portion of the order numbers varies with product releases and manual editions. For Software AG manuals, "vrs" is the version, revision, and system maintenance (SM) level of the product for which the manual was revised or updated.

1

SUPPORTED ENVIRONMENTS

Before attempting to install Adabas, ensure that the host operating system is at the minimum required level.

Adabas version 7.4 is available for BS2000 OSD 2.0 and above operating system environments.

Software AG provides Adabas support for the operating system versions supported by their respective manufacturers. Generally, when an operating system provider stops supporting a version of an operating system, Software AG will stop supporting that operating system version.

Although it may be technically possible to run a new version of Adabas on an old operating system, Software AG cannot continue to support operating system versions that are no longer supported by the system's provider.

If you have questions about support, or if you plan to install Adabas on a release, version, or type of operating system other than those included in the list above, consult your Adabas technical support facility to determine whether support is possible, and under what circumstances.

2

BS2000 SYSTEMS INSTALLATION

This chapter describes the preparation for and installation of Adabas on systems running under the Siemens BS2000 operating system (see page 5 for specific BS2000 systems supported by this version of Adabas).

Information for your specific installation is contained in the *BS2000 Product Installation Package* (order number SIA-111-016) distributed with the System Installation Aid.

Installation Checklist for BS2000

The following list provides an overview of the Adabas installation procedure on BS2000 systems. The remainder of the chapter provides specific information for the installation process.

1. Provide disk space for the Adabas libraries.

   The libraries are restored from the installation tape.

2. Allocate disk space for the Adabas database.

   For better performance, distribute the database files over multiple devices and channels.

3. Define the address space for running the Adabas nucleus.

4. Allocate and format the Adabas database with the ADAFRM utility job (job I030, step 1000).

5. Define the global database characteristics with the ADADEF utility job (job I030, step 1000).

6. Start the Adabas nucleus and test the Adabas communications with the ADANUC job (job I040, step 1000).

7. Load the demonstration (demo) files ADAvrs.EMPL/MISC/VEHI with the ADALOD utility. (performed in Job I050 steps 0001 to 0003).

8. If appropriate, test Adabas address space communications by running ADAREP (Job I050 step 9990).

9. If appropriate, load the Adabas Online System (AOS) selectable unit into a Natural system file by running the AOSINPL job. Alternatively, install the AOS demo version delivered with Adabas (job I061, step 0112).

10. Terminate the Adabas nucleus with an ADAEND operator command

11. Back up the database by running the ADASAV utility job.

Preparing to Install Adabas

Before you install Adabas in a BS2000 system

- Define a BS2000 logon ID for Adabas that permits loading from a release or installation tape;
- Enable all privileges for the logon ID to allow optional operations. This includes such capabilities as priority, T/P class, and maximum virtual memory size.

  When allocating direct access files, private volumes are preferred to avoid the space fragmentation that can occur with public volumes.

Adabas Library Disk Space Requirements

The Adabas files and libraries require the following disk space in PAM pages where “vrs” is the Adabas version/release/system maintenance (SM) level:

File Name	Size	Description
ADAvrs.MOD/	1728	Module library
ADAvrs.SRC	480	Adabas source, macros, and example jobs
WALvrs.SRC	288	Entire Net-Work source, macros, and example jobs
WALvrs.MOD	384	BS2000-dependent modules for Entire Net-Work
AUTvrs.LIB	039	ADAUTM library (LMS) (optional)

Disk Space Requirements for the Database

The Adabas database size is based on user requirements. For more information, refer to the *Adabas DBA Reference Manual*. The following are suggested 2000 device cylinder and PAM page sizes for an initial Adabas database, allowing for limited loading of user files and the installation of Natural:

Database Component	Cylinders	PAM Pages
ASSOR1 (Associator)	50	4000
DATAR1 (Data Storage)	250	20000
WORKR1 (Work space)	50	4000
TEMPR1 (temporary work space)	25	2000
SORTR1 (sort work space)	25	2000

Adabas Nucleus Partition/Address Space Requirements

The typical Adabas nucleus requires at least 800-1024 KB of storage to operate. The size of the nucleus partition address space may have to be larger, depending on the ADARUN parameter settings. The address space parameters for the BS2000 user ID's JOIN entry must also be adequate. Parameter settings are determined by the user.

Using DAB-Supported Volumes

The following restrictions apply to Adabas components when located on DAB-supported volumes:

ASSO	not recommended for read caching; prohibited for write caching.
DATA	not recommended for read caching; prohibited for write caching.
WORK	no restrictions for read caching; prohibited for write caching.
CLOG/PLOG/RLOG	no restrictions for read caching; prohibited for write caching.
SORT/TEMP	no restrictions.
Sequential input	no restrictions.
Sequential output	no restrictions.

Migrating an Existing Database

Use the ADACNV utility to migrate existing databases to new releases of Adabas. See the *Adabas Utilities Manual* for more information.

Installing the Adabas Release Tape

Adabas release tapes available to Software AG affiliates contain all Adabas product options. The affiliates use these release tapes to create custom installation tapes for customers according to contract agreements.

For specific information about your particular release tape, refer to the *Installation Notes* delivered with the installation tape.

SMA Installation

If you are installing Adabas using the Software AG System Maintenance Aid (SMA), refer to the *System Maintenance Aid Manual* and to the information provided with the installation tape for specific installation instructions.

Manual Installation

If you are not using SMA, copy the datasets from tape to disk using the procedure described below.

1. Copy the Library SRV*nnn*.LIB from Tape to Disk

This step is not necessary if you have already copied the library SRV*nnn*.LIB from another Software AG tape. For more information, refer to the element #READ-ME in this library.

The library SRV*nnn*.LIB is stored on the tape as the sequential file SRV*nnn*.LIBS containing LMS commands. The current version *nnn* can be obtained from the Report of Tape Creation. To convert this sequential file into an LMS-library, execute the following commands:

```
/IMPORT-FILE  SUPPORT=*TAPE(FILE-NAME=SRVnnn.LIBS, -
/   VOLUME=<volser>, DEV-TYPE=<tape-device>)
/ADD-FILE-LINK  LINK-NAME=EDTSAM, FILE-NAME=SRVnnn.LIBS, -
/   SUPPORT=*TAPE(FILE-SEQ=3), ACC-METH=*BY-CAT, -
/   BUF-LEN=*BY-CAT, REC-FORM=*BY-CAT, REC-SIZE=*BY-CAT
/START-EDT
@READ  '/'
@SYSTEM 'REMOVE-FILE-LINK  EDTSAM'
@SYSTEM 'EXPORT-FILE  FILE-NAME=SRVnnn.LIBS'
@WRITE  'SRVnnn.LIBS'
@HALT
/ASS-SYSDTA  SRVnnn.LIBS
/MOD-JOB-SW  ON=1
/START-PROG  $LMS
/MOD-JOB-SW  OFF=1
/ASS-SYSDTA  *PRIMARY


<tape-device>    = device-type of the tape, e.g. TAPE-C4
<volser>         = VOLSER of tape (see Report of Tape Creation)
```

2. Copy the Procedure COPY.PROC from Tape to Disk

Call the procedure P.COPYTAPE in the library SRV*nnn*.LIB to copy the procedure COPY.PROC to disk:

```
/CALL-PROCEDURE  (SRVnnn.LIB,P.COPYTAPE), -
/   (VSNT=<volser>, DEVT=<tape-device>)
```

If you use a TAPE-C4 device, you can omit the parameter DEVT.

3. Copy all Product Files from Tape to Disk

Enter the procedure COPY.PROC to copy all Software AG product files from tape to disk:

```
/ENTER-PROCEDURE  COPY.PROC, DEVT=<tape-device>
```

If you use a TAPE-C4 device, you can omit the parameter DEVT. The results of this procedure are written to the file 'L.REPORT.SRV'.

The Adabas BS2000 Communication Environment

The installation of a supervisor call (SVC) to provide a communication environment for Adabas is not required on BS2000 systems. Adabas uses the BS2000 executive services' "common-memory pool" and "eventing" for interprocess communication.

The router functions are implemented in the form of subroutines contained in the module ADARER. ADARER is loaded into common memory pool during nucleus initialization, and is shared by all user tasks that issue Adabas calls.

Applying ZAPs

Every effort has been made to make Adabas operating-system-independent; however, Adabas development is done in an IBM environment and corrections and changes are prepared in this environment.

As a result, corrections are applied to Adabas modules with IMASPZAP, which determines the syntax of any ZAPs. When corrections must be applied to a module library at BS2000 locations, the ZAPs must be revised to the LMS format. The following example shows how this can be done:

1. Assuming a ZAP in the following LMS format:

```
/EXEC LMS
LIB ADAvrs.MOD,BOTH
UPDR ADAEXAMP          (a)
*COR ADAEXAMP,12BC,X'47B0A123'=X'4720BAEE'      (b,c)
*END
END
```

—where:

(a) specifies corrections to module ADAEXAMP.

(b) verifies the contents at location X'12BC'. The correction is applied only if the content of this location is X'47B0A123'.

(c) specifies a replacement value for the location X'12BC' of X'4720BAEE'.

2. To verify the original data at location X'12BC' on the BS2000 system, perform the following:

/LOAD (ADAEXAMP,ADAvrs.MOD)	Load module
/DISPLAY L'12BC'.(L=4)	Display original content
%D C=ADAEXAMP.H'12BC'%XL4	(AID)

If the data displayed is different from the value given on the original VER statement, do not continue with step 3.

3. Run the LMS procedure.

4. To ensure that the last step was performed properly, perform step 2 again. This procedure should also be used for ADARUN and ADALNK ZAPs, described later in this chapter.

Job Variable (JV) Handling

The control job variable (JV) is filled automatically if the statement

/DEL-JV name
/SET-JOB-STEP
/CRE-JV name
/SET-JV-LINK *ADA,name

—is contained in the nucleus start-up job control statements. The JV layout is as follows:

Positions	Length	Representation	Meaning
1 - 128	128	undefined	unused
129 - 136	8	alphanumeric	program name
137 - 140	4	numeric	user ABEND code
141 - 145	5	numeric	error number (parameter or utility)
146 - 150	5	numeric	Adabas response code
151 - 157	7	numeric	CPU time, program-related (sec.)
158 - 158	1	alphanumeric	blank
159 - 165	7	numeric	elapsed time, program-related (sec.)
166 - 178	13	alphanumeric	blank, reserved

Job Switches

Adabas uses job switch 10. The job switch is set by ADARUN and reset if the nucleus or utility session terminates normally.

It can be used for session control to indicate whether a termination is normal.

When ADARUN is called with PROGRAM=USER, no switches are set or reset.

Linking User Exits A and B to ADALNK

One or two user exits may be linked with ADALNK:

- UEXITB (pre-command) receives control **before** a command is passed to a target by the router 04 call.

  Notes:
  Special commands emanating from utilities and from Adabas Online System are marked as "physical" calls. These calls must be bypassed in user exits. These calls have X'04' in the first byte (TYPE field) of the command's Adabas control block (ACB). UEXITB must check this byte and return if it is set to X'04'. Be sure to reset R15 to zero on return.

- UEXITA (post-command) receives control **after** a command has been completely processed by a target, the router, or by ADALNK itself.

  At entry to the exit(s), the registers contain the following:

Register	Content
1	Address of the UB. If the flag bit UBFINUB is reset, the contents of the halfword at Adabas + X'86' have been moved to UBLUINFO. If those contents are greater than zero, the two bytes starting at UBINFO (UB+X'40') have been set to zero. If UBFINUB is set, no changes can be made to the UB or ACB (except for ACBRSP).
13	Address of an 18-word save area
14	Return address
15	Entry point address: UEXITB or UEXITA

Any registers except register 15 that are modified by the user exits must be saved and restored; the address of a save area for this purpose is in register 13.

If at return from UEXITB register 15 contains a value other than zero, the command is not sent to the target but is returned to the caller. The user exit should have set ACBRSP to a non-zero value to indicate to the calling program that it has suppressed the command: response code 216 is reserved for this purpose.

The UEXITB exit may set the UB field UBLUINFO to any lesser value, including zero; an ABEND occurs if the user exit sets UBLUINFO to a greater value. The UBLUINFO length cannot be changed when any other exit is used; for example, Adabas Review.

The user information received by a UEXITA exit may have been modified; this modification may include decreasing its length, possibly to zero, by any of the ADANUC user exits.

Connecting UES-Enabled Databases

Prior to Adabas version 7, Entire Net-Work converted all data for mainframe Adabas when necessary from ASCII to EBCDIC. Starting with version 7, Adabas is delivered with its own data conversion capability called universal encoding support (UES). Entire Net-Work detects when it is connected to a target database that converts data and passes the data through to Adabas without converting it.

For Adabas version 7.4, UES is enabled by default for the link routine ADALNK.

Notes:
The use of UES-enabled link routines is transparent to applications, including applications that do not require UES translation support: it is not necessary to disable UES support.

Load Module

There is only one load module for ADALNK on BS2000 and this has to be linked with LNKUES and the default translation tables. LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

The two standard translation tables are:

- ASC2EBC: ASCII to EBCDIC translation; and

- EBC2ASC: EBCDIC to ASCII translation.

  The Adabas translation table pair is provided in Appendix A.

Default or Customized Translation Tables

You may use the load modules with the default translation tables linked in, or you may prepare your own customized translation tables, reassemble the tables, and link them with the LNKUES module that is delivered.

Notes:
It should only be necessary to modify these translation tables in the rare case that some country-specific character other than A-Z a-z 0-9 must be used in the additions 1 (user ID) or additions 3 field of the control block.

The LNKUES module is functionally reentrant; however, it is not linked that way in the Adabas load library.

Source Modules

The ADALNK module has been coded to enable UES support is accessible via weak external references and will be enabled if the LNKUES module is linked or bound to it.

Job Steps

Job library member ALNKUES is an example job of how to link ADALNK with the UES components.

Calling LNKUES

LNKUES is called only on ADALNK requests (X‘08’), (X‘0C’) with reply (X‘10’) or (X‘1C‘) and reply (X‘20’) calls if the first byte of the communication ID contains X‘01’ and the second byte does not have the EBCDIC (X‘04’) bit set.

For requests, LNKUES receives control before UEXITB. For replies, LNKUES receives control after UEXITA.

Required Environment

The Adabas database must be UES-enabled. See the *Adabas DBA Reference Manual* and the ADACMP and ADADEF utility chapters in the *Adabas Utilities Manual* for more information.

Connection Possibilities

UES-enabled databases are connected to machines with different architectures through Smarts, or through Entire Net-Work.

In the case of Smarts, Adabas SQL Server (AQA) clients may not be strictly EBCDIC in an environment where databases are connected through Software AG's internal product Smarts (APS); therefore, the Adabas link routine ADALNK contains an entry ADALCO.

For Smarts and Net-Work the ADALNK will be UES-enabled by linking LNKUES, ASC2EBC and EBC2ASC using the sample jobstream to link the ADALNK module in this way is ALNKUES.

Linking LNKUES to ADALNK for Data Conversion

Adabas version 7 is delivered with the module LNKUES for Universal Encoding Support (UES). This module must be linked to ADALNK. LNKUES converts data in the Adabas buffers and byte-swaps, if necessary, depending on the data architecture of the caller.

Prior to version 7, Entire Net-work converted all data for mainframe Adabas. When Entire Net-work version 5.5 and above detects that it is connected to a target database that converts data, it passes the data through without converting it.

LNKUES is called only on ADALNK request (X'1C') and reply (X'20') calls if the first byte of the communication ID contains X'01' and the second byte does not have the EBCDIC (X'04') bit set.

- For requests, LNKUES receives control before UEXITB.
- For replies, LNKUES receives control after UEXITA.

By default, two translation tables are linked into LNKUES/ADALNK:

- ASC2EBC: ASCII to EBCDIC translation; and

- EBC2ASC: EBCDIC to ASCII translation.

  Notes:
  It should only be necessary to modify these translation tables in the rare case that some country-specific character other than A-Z a-z 0-9 must be used in the additions 1 (user ID) or additions 3 field of the control block.

  If you prefer to use the same translation tables that are used in Entire Net-Work:

- in ASC2EBC and EBC2ASC, change the COPY statements from UES2ASC and UES2EBC to NW2ASC and NW2EBC, respectively.
- reassemble the translation tables and relink LNKUES/ADALNK.

  Both the Adabas and Entire Net-Work translation table pairs are provided in appendix B. You may want to modify the translation tables or create your own translation table pair. Be sure to (re)assemble the translation tables and (re)link LNKUES/ADALNK.

```
/ASS-SYSDTA *SYSCMD
/STA-PROG $TSOLNK
MODULE ADALNK,LIB=USER.MOD,ELEM=ADALNK
NCAL
LINK SYMBOLS *KEEP
INCLUDE ADALNK,ADABAS.MOD
INCLUDE LNKUES,ADABAS.MOD
INCLUDE ASC2EBC,ADABAS.MOD
INCLUDE EBC2ASC,ADABAS.MOD
BIND
/ASS-SYSDTA *PRIM
```

  The (re)linked ADALNK must be made available to Entire Net-Work. If you are calling Adabas version 7 and you do not have the correct LNKUES/ADALNK module, Adabas produces unexpected results: response code 022, 253, etc.

Formatting New Adabas Datasets

The following formatting is required when creating new datasets:

Dataset	Format...
ASSO	the first track plus the first 30 RABN blocks
DATA	the first two tracks
WORK	the whole dataset
PLOG / CLOG / RLOG	the whole dataset
SORT / TEMP	(no formatting required)

Formatting SORT and TEMP

The SORT and TEMP files can optionally be formatted. If non-formatted files are used, the following two FILE commands must be issued:

```
/CREATE-FILE ADA99.SORT,PUB(SPACE=(4800))
/SET-FILE-LINK DDSORTR1,ADA99.SORT,BUFF-LEN=STD(2),OPEN-MODE=OUTIN
```

Starting Adabas

User Exit 2 Start Options

Adabas user exit 2 for BS2000 permits two methods for specifying the /ENTER-JOB job:

- Using a job variable containing the complete /ENTER-JOB job command;
- Defaulting to an ENTER-JOB RES.E.xLCO start-up command.

For more information, see the description of user exit 2 in the *Adabas DBA Reference Manual*.

Interpreting BS2000 Error Messages

When running Adabas, some BS2000 messages may indicate that a parameter is missing or is incorrect. The following are the message IDs that can occur, and their explanations in an Adabas environment:

D922	SAM file allocation (for example, for DDOUT1) primary and secondary allocation must be a multiple of the both standard block size (16) and of the allocation unit (3). The smallest multiple allowed is 48. The standard block size may have been changed using the ADARUN parameter QBLKSIZE. If so, the standard block size is determined by dividing the sum of the QBLKSIZE value and 2047, by 2048.
DD99	Parallel access to a PAM file (for example, DATA) was attempted, but SHARUPD=YES was omitted.
DDB1	(refer to the D922 description, above.)
DDC2	SHARUPD=YES was specified for a SAM file (for example, DDSIBA).

3

INSTALLING ADABAS WITH TP MONITORS

This chapter provides information needed to install Adabas with teleprocessing (TP) monitors TIAM and UTM. Information about using Adabas with TP monitors is also contained in the installation chapter.

The Adabas API for BS2000

The Adabas version 7 Adabas application programming interface (API) comprises the modules

ADAUSER (delivered in source)
ADAUSER2 (delivered in source)
ADALNK (delivered in source)
ADAL2P
SSFB2C (delivered in source, plus macro B2CONFIG to customize the interface)

ADALNK contains the combined functionality of all the ADALNx modules of previous Adabas versions. Software AG recommends that you link ADAUSER to the application program and use the entry point ADABAS on the API call.

For compatibility with existing applications, ADALNK contains the following entry points:

Entry Point	TP Monitor
ADALNK	Batch / TIAM
ADALNR	Batch / TIAM (reentrant)
ADALNN	UTM running Natural
ADALNU	UTM with Assembler or a 3GL language
ADALNQ	Adabas SQL Server (ESQ)

ADALNK and ADAL2P are reentrant. ADAUSER, which is supplied as source to allow modification, can be made reentrant if the user application provides a 4-byte anchor area.

Because ADALNK is reentrant, parameter items previously moved into ADALNK are now found in SSFB2C.

The symbol “ADABAS” is no longer in ADALNK, but in ADAUSER.

The symbol “ADALNR” is offset X’C’ (decimal 12) bytes into the start of ADALNK. If fields are to be accessed in this version of ADALNK, their offsets have changed accordingly.

ESD Symbols Used in the Adabas API

The following symbols have a special meaning in the Adabas API:

Symbol	Meaning
ADAUSRID	identification area used by Natural; identifies the caller as Natural
AUTUSRID	identification area used by ADAUTM
CMSTART	identifies the batch Natural driver
KDCKB	area in UTM holding the user/task ID
KDCUTMD	identifier that the carrier is UTM
KDCUTMID	alternative identifier that the carrier is UTM

Software AG advises you not to define or write-access any of these symbols in applications that also use the Adabas API.

ADAUSER

ADAUSER is a program that links the user to Adabas. It is specific to an operating system and is independent of release level and mode. It can be used in batch and in some TP environments.

Software AG recommends that you link ADAUSER to your applications because it

- can be made reentrant;
- is small; and
- has no version-specific functions. In future versions, application link jobs will not require changes. For example, the application and its linkage will not be affected if TCP/IP is used to access the database.

ADAUSER propagates symbols to the loaded API interface, which can be used to pass user data or determine the carrier system the application is using. ADALNK does not do this. If ADALNK is to be linked, the whole API must be linked to the application.

ADAUSER Loading ADALNK

The following file statement assigns a SAM/V dataset for the Adalink parameters:

```
/SET-FILE-LINK DDLNKPAR,samv-dataset
```

If the file link DDCARD is specified and the application is **not** running on UTM, ADAUSER loads ADARUN. This is the traditional API.

If the file link DDCARD is **not** present, ADAUSER loads ADALNK. In this case, the routing information (idt_name,dbid) and the Adabas Review buffer size can be defined in the file link DDLNKPAR.

ADAUSER Loading ADARUN

If the file link DDLNKPAR is **not** specified, ADAUSER loads ADARUN for using prefetch and multifetch. The routing and Adabas Review buffer parameters are then to be delivered in the file link DDCARD or *SYSDTA.

The following file statement assigns the proper Adabas module library “modlib”:

```
/SET-FILE-LINK DDLIB,modlib
```

ADAUSER can load modules from several libraries. If the ADARUN-defined library is found in the catalog, it becomes the primary library.

On the first call to Adabas, ADAUSER loads the latest version of ADARUN. This makes the calling process release-independent. Subsequent Adabas calls bypass ADARUN.

ADARUN processes its control statements:

- If ADARUN PROGRAM=USER (the default) and ADARUN MODE=MULTI (the default), ADARUN also loads ADALNK.
- If ADARUN PROGRAM=USER (the default) and ADARUN MODE=SINGLE, ADARUN also loads ADANUC.

This makes ADAUSER mode-independent.

ADAUSER2

ADAUSER2 is a program that allows Natural to load a COBOL module where Natural and COBOL have separate Adabas session IDs. ADAUSER2 is delivered in the ADAvrs.SRC library and must be assembled before it can be used.

Fixed Linking of ADALNK or ADAUSER

To resolve external symbols in the application module, link the following modules to it:

- for ADALNx symbols, ADALNK, ADAL2P, and SSFB2C
- for the ADABAS symbol, ADAUSER (note that the delivered ADAUSER is not reentrant)

If you are linking ADALNK, ADAL2P, and SSFB2C to the application module **and** you need to resolve the ADABAS symbol as well, you can alternatively do one of the following:

1.. Insert the TSOSLNK command RENAME in the application link job to rename ADALNK before you include it:

```
RENAME ADALNK,ADABAS
INCLUDE ADALNK,$SAG.ADABAS.MOD
```

2.. To avoid changing the link job, insert the following lines behind the label ADALNU in the supplied ADALNK source in the ADAvrs.SRC library:

```
        ENTRY ADABAS
ADABAS DS     0F
```

Then reassemble the ADALNK using the macros supplied in the ADAvrs.MAC library and link it to the application.

Routing and Adabas Review Parameters

Routing parameters are used to locate the application's target database:

```
DBID=
IDTNAME=
```

If the application requires an Adabas Review buffer, the following parameter is used:

```
LRVINFO=256
```

If ADALNK has been linked or loaded, these parameters are found under the link name DDLNKPAR. ADAL2P reads the data from DDLNKPAR and holds it in SSFB2C.

If ADAUSER has been linked or loaded and

- the file link DDCARD is present, ADARUN is loaded and these parameters are read from DDCARD. This is the traditional Adabas API.
- the file link DDCARD is **not** present, ADALNK is loaded and these parameters are found in DDLNKPAR.

ADARUN must be loaded in order to load and install the Adabas prefetch components:

```
ADARUN PROG=USER,PREFETCH=YES
```

Presetting Parameters in SSF2BC

Routing, Adabas Review, and some configuration information can be preset in SSFB2C using the delivered source module in ADAvrs.SRC and the macro B2CONFIG from ADAvrs.MAC:

SSFB2C Parameter	Description
IDTNAME=ADAxxxxx	ID table where the database runs
LRVINFO=256	set up Adabas Review buffer
X48=N	trun off the Natural version 2.3 IPC X‘48’ call logic

For example, the following presets routing to database 99 on the IDTNAME ADA00009:

```
SSFB2C CSECT
  B2CONFIG DBID=99,IDTNAME=ADA00009
  END
```

ADALNK

When loaded, ADALNK attempts to read and process control statements from its parameter dataset. Entries in the parameter dataset appear in the following format:

```
ADALNK  IDTNAME=ADA12345
ADALNK  DBID=17
```

The ADALNK control statements are delivered with the default values specified in the following discussion. Changing default settings by using ZAPs or by reassembling ADARUN and ADALNK should be the **rare exception**.

Note:
In a future version of Adabas, it will not be possible to change ADARUN and ADALNK parameters with ZAPs.

ADALNK Parameters

Important:
Do not specify ADALNK statements in ADARUN nuclei, Entire Net-work, ADAUSER, or utility contexts. If you do, a warning message (ADAK06) is reported, the ADALNK statements are ignored, and processing continues.

DBID= {nnnnn |1 }

Sets the default database ID. The minimum value is 1; maximum value is 65535. The default is 1.

DISACLOS={ YES | NO }

Determines whether Adalink closes all communication after executing the CL command to the last connected database so that the user task can issue BS2000 WRCPT macros (not available for ADALNN, ADALNR, or ADALNU Adalinks). The default is NO.

GROUPS={ YES | NO }

Determines whether the communication items are valid for all logon user IDs with no restriction (NO, the default) or whether they are valid only for the logon user ID of the present task (YES). See also the ADARUN GROUPS parameter in the *Adabas Operations Manual*.

IDTNAME={ ADABAS5B |ADAccccc }

Specifies the name of the ID table to be accessed. This name must be the same as the nucleus' ADARUN IDTNAME parameter, if specified, which must be 8 characters beginning with "ADA". The default is ADABAS5B.

LRVINFO={256 | 0 }

Determines whether the Review user exit B is inactive (0, the default) or active (256). Values other than 0 and 256 are not allowed.

X48={ YES | NO }

Determines whether X48 call processing is available (YES, the default) to an application so that it can build its own 28-byte communication ID to identify internal Adabas resources used by a logical transaction.

ADALNK Parameter Service

The ADALNK parameter service coordinates ADARUN, ADALNK, and SSF parameters. It comprises three modules:

- ADAL2P contains the parameter service routines
- the ADALNK module contains the Adalink parameters
- SSF2BC contains operating-system-specific parameters. Most of the operating-system-related defaults (such as the names, scopes, and sizes of memory pools, serialization and event items) are concentrated in this module.

  The parameters in the delivered SSF2BC module contain the same defaults as associated ADARUN and ADALNK parameters and can be modified with the help of the ADARUN or ADALNK parameters. In some cases, it is desirable to “hide” these parameters.

Linking ADALNK

The ADALNK parameter service considers the SSFB2C configuration module:

- If ADALNK is loaded by ADAUSER, SSFB2C is AUTOLINKED by V-Constant. In this case, no changes are required.
- If ADALNK needs to be bound to a user program, include the following statements in the TSOSLNK JCL that links ADALNK:

```
...
INCLUDE ADALNK
INCLUDE ADAL2P
INCLUDE SSFB2C
or BINDER JCL :
...
INCLUDE-MODULES   ELEMENT=(ADALNK,ADAL2P,SSFB2C),-
...
```

If the INCLUDE statement for ADAL2P is omitted, an unresolved VCONS occurs and the parameter service will not be available.

B2CONFIG Macro

If other **defaults** are required, the SSF configuration module can be reassembled. The macro B2CONFIG is delivered for this purpose.

The following operands are available:

ALNKPRT={ NO | YES }

Permits or inhibits the creation of the ADALNK parameter services' protocol dataset.

DBID= {nnnnn |1 }

Sets the default for the ADARUN and ADALNK parameter DBID=n. The maximum value is 65535; the default is 1.

DISACLOS={ YES | NO }

Sets the default of the ADALNK parameter DISACLOS.

ENVNAME=cccccccc

Sets the default for the ADARUN and ADALNK parameter IDTNAME=. The operand must comprise exactly 8 alphanumeric characters.

ENVSCOPE= {GROUP | GLOBAL }

Sets the default for the ADARUN and ADALNK parameter GROUPS. Specify

- ENVSCOPE=GROUP for ADALNK GROUPS=YES
- ENVSCOPE=GLOBAL for ADALNK GROUPS=NO.

LRVINFO={256 | 0 }

Sets the default for the Adalink parameter LRVINFO.

NUMGES={ nnn | 6000 }

Determines the maximum number of Global Event Control Blocks allowed in the machine.

Parameter Priority

Parameter values changed by ZAPs take **priority** over the corresponding SSFB2C.

Parameter values set by ADARUN or ADALNK statements take priority over both values changed by ZAPs and values from SSFB2C.

Parameter values changed by ZAPs are not reported on the protocol dataset.

ADALNK Protocol Dataset

The ADALNK parameter service allows you to protocol its statements

- into SYSLST (ASS-SYSLST dataset);
- into a dataset determined by "/SET-FILE-LINK DDPLNPRT,dataset" (if this link name is not included in the tasks file table, the protocol is written to SYSLST); or
- nowhere. In this case, the module SSFB2C must be assembled by setting the operand ALNKPRT=NO.

New Messages

```
ADAK04   CONFIGURATION MODULE FOUND :
ADAK04   NAME : USERCONF; ASS-DATE 960229; VERSION 0130        note 1)
ADAK04   ENVNAME  =ADATEST1                                    note 2)
ADAK04   ENVSCOPE =GLOBAL                                      note 2)
ADAK04   DBID     =145                                         note 2)
ADAK04   LRVINFO  =0                                           note 2)
ADAK04   DISACLOS =NO                                          note 2)
ADAK04   THE FOLLOWING ADALNx PARAMETERS ARE IN USE FOR THIS RUN
ADALNK DBID=196                                                note 3)
```

Notes:

1. *Information to identify a user-defined configuration module.*
2. *B2CONFIG macro parameters.*
3. *ADALNK parameter read from the ADALNK parameter service.*
4. *Values changed by ZAPs are not reported.*

Assembling ADALNK

The source form of ADALNK is in the Adabas source library. The procedure

/CLP ADAvsn**.P,(ASM,ADALNK,LIB=USER.MOD)**

—assembles the file S.ADALNK into the USER.MOD library.

The following are the selectable options available when reassembling an Adalink module:

Option	Default	Action Required
LOGID	1	Specify a default logical database ID ranging 1-65535
LUINFO	0	Specify a user data length for the data passed from the ADALNK to Adabas user exit 4.

Binding/Linking ADALNK above 16 MB

If you intend to link or bind ADALNK into contexts located above the 16-megabyte line, the ADALNK characteristics must be set to AMODE=31 and RMODE=ANY.

Installing Adabas with Batch / TIAM

The Adalink used for batch and the TIAM TP monitor environment under BS2000 is ADALNK.

The ADALNK entry point is used where a single user issues only one Adabas call at a time and waits synchronously on the call to return. The last eight bytes (UID) of the communication ID are set to either

- "Bxxxx" (batch); or
- "Dxxxx" (TIAM)

—where "xxxx" is the BS2000 task sequence number.

This Adalink is loaded by ADARUN to control multiuser (ADARUN MODE=MULTI) or utility sessions. The CSECT name is "ADALNK".

ADALNK for SAP® Application Packages

The ADALNR entry point is located at offset xC in ADALNK, which is reentrant.

Calls to ADALNR require an eighth Adabas call parameter containing the address of this initialized area, which must be below the 16-megabyte line.

The seventh parameter, which is reserved for Natural applications, must always be defined. This parameter must always be used for each Adabas call.

The addressed area itself, which cannot be changed by the user, must equal the total of the two halfword counts found at ADALNK locations X'B4' and X'B6'.

Dynamically Loading Symbols in Batch / TIAM

The binder/lader/starter (BLS) may be used to dynamically load the ADALNx (where "x" is K, N, Q, R, or U) or ADABAS symbol with the job statements

```
/SET-FILE-LINK DDLIB,<adabas_library>
.
/START-PROGRAM (MY.LIB,MYAPPL),RUN-MODE=ADVANCED(ALT-LIB=YES)
.
```

—where "adabas_library" is the name of the delivered Adabas module library.

If the program MYAPPL has the unresolved external symbol

- ADALNx, then ADALNK, ADAL2P, and SSFB2C are loaded.
- ADABAS, then ADAUSER is loaded, which in turn loads ADARUN, ADAIOR, etc.

Installing Adabas with UTM

Operation Options

UTM can operate with Adabas in two modes:

- **Synchronized**. UTM transactions and database transactions are coordinated.

  UTM is aware of database transactions and coordinates its transactions with Adabas, providing automated restart in case of failure. The selectable unit ADAUTM documented in appendix C starting on page 57 is required to implement this option.

- **Unsynchronized**. The application completes database transactions independently of UTM.

  UTM is not involved in the database transactions. Several Adabas transactions can occur within a single UTM transaction. Applications call Adabas from the ADALNK module directly. The following sections describe the process of selecting the correct UTM ADALNx entry point.

Unsynchronized Operation

UTM conforms to the KDCS (compatible data communication interface) description, which requires a TP program with the following general sequence:

1 initialize
2 obtain the terminal input data
3 process the data (including any Adabas calls)
4 write the output data to the terminal
5 end (PEND).

Under UTM, a BS2000 task processes only a single user until PEND. By defining multiple UTM tasks for an application, requests are processed in parallel, thus improving performance.

Multiple tasks are also required to prevent deadlock situations. For example, if only one UTM task is available and user 2 requests a record held by user 1, all processing stops (deadlock) until user 2 is timed-out, thus freeing the UTM task to complete user 1’s transaction.

Running ADASAV under UTM

The Adabas utility ADASAV should only be run on a UTM system when very few update transactions are active, or “deadlock” can result. This occurs when fewer UTM tasks have been defined than can accommodate the number of Adabas user transactions currently open, and the ADASAV synchronization begins.

When ADASAV performs synchronization, all ET transactions are held in the command queue (CQ) until none remain to be processed. If there are more open UTM transactions than UTM tasks available, other transactions cannot be completed until the UTM transactions end; this results in deadlock.

All transactions remain frozen, waiting for time-out to occur. When the first time-out of a UTM transaction occurs, a UTM task is freed and another UTM transaction takes its place. This continues until all frozen transactions are freed, bringing synchronization to an end.

—where “adabas_library” is the name of the delivered Adabas module library.

UTM Adalink Entry Points

Software AG strongly recommends that you link ADAUSER to the UTM application and that you use the ADABAS entry point.

For compatibility reasons, ADALNK may be linked to a UTM application using one of the following entry points:

- ADALNN for systems running with Natural, and
- ADALNU for systems running without Natural.

Both are described as provided on the Adabas/BS2000 release tape; however, ADALNN and ADALNU defaults can be changed (zapped). While the Adabas call is being processed, the UTM task waits synchronously until the Adabas nucleus returns the call results.

ADALNN

ADALNN is used where the calling program works with different users, or must identify more than one transaction. However, only one Adabas call is processed at a time, and waits are synchronous. ADALNN is used with Natural/UTM or Natural/TIAM/MULTIPASS. The CSECT name is ADALNN. There are no additional ENTRYs.

ADALNU

ADALNU is for UTM applications that do not run with Natural. The UID part of the communication ID is set with the logical terminal ID derived from the KB header field KCLOGTER. For UTM version 3, the KB is found by the weak external KDCKB that is satisfied in KDCROOT. For older UTM versions, or in environments where ADALNU is not linked statically to KDCROOT, ADALNU locates the KB in the program's parameter list by going back up the save area chain to the KDCROOT save area.

Linking ADAUSER to UTM Applications

Natural

Software AG recommends that you link ADAUSER to the Natural UTM application and set the Natural driver parameter

ADACALL=NO

ADAUSER detects that UTM is the carrier system. It loads ADALNK directly and thus reads its parameters from DDLNKPAR.

COBOL or Assembler

If ADAUSER is linked to a COBOL or Assembler UTM application, ADALNK is loaded and parameters are read from DDLNKPAR.

Loading ADALNK

Whenever you link ADAUSER to UTM applications, you need to add the following link statement to the UTM start job to load the ADALNK:

```
/SET-FILE-LINK DDLIB,<adabas_library>
```

4

DEFINING NEW DEVICE TYPES

Support for new device types that include user-defined block sizes can be implemented in Adabas by modifying one of the table of device-constant entries (TDCEs) reserved for this purpose. A TDCE is X'40' bytes long and the first free TDCE can be identified by X'0000' in its first two bytes (TDCDT).

Under BS2000, the address of the first TDCE is at offset ADAIOR+ X'34' for all versions of Adabas.

Adabas direct access datasets are always mapped to UPAM files, removing the need to consider physical device characteristics. PAM pages in a dataset are addressed relative to the dataset beginning.

Adabas blocks comprise one or more PAM pages. An Adabas virtual track is made up of a fixed number of blocks, and an Adabas virtual cylinder comprises a fixed number of tracks. The definition of tracks and cylinders are independent of the physical device.

There are a number of predefined virtual devices for BS2000 that should meet most of the storage capacity needs that arise (refer to the *Adabas Operations Manual*). It should be noted that the virtual memory requirement increases significantly with a larger block size.

Support for new device types, including user-defined block sizes, can be implemented in ADAIOR by modifying one of the TDCEs reserved for this purpose.

Information to be Zapped into the First Free TDCE

The information in the following tables must be zapped into the first free TDCE. The rules described in the section **General Rules for Defining Device Block Sizes** starting on page 40 must be followed when changing the TDCE.

Label	Offset	Contents
TDCDT	00	Device type in unsigned decimal (X'3385'), must be numeric, and unique among all TDCEs.
TDCKSN	02	Constant set number: must be uniquely chosen from the values X'28' (reserved for BS2000 device type 2006), X'2B', or X'2E'.

Label	Offset	Contents
TDCF	03	The flag bit must be set—TDCFFBA (X'80') for FBA/PAM devices or TDCFCKD (X'40') for CKD devices.
TDCDT1	04	Set to zero under BS2000.
TDCDT2	05	Set to zero under BS2000.
TDCDT3	06	Set to zero under BS2000.
TDCDT4	07	Set to zero under BS2000.
TDCMSBS	08	In BS2000, 32760 for compatibility. Refer to the TDCMSBS default table in appendix A of the *Adabas Operations Manual* for more system- and device-related information.
TDCTPC	0A	Number of tracks per cylinder.
TDCCIPT	0C	Number of FBA blocks or PAM pages per track (if TDCFFBA is set). For BS2000 less than or equal to 16.
TDCBPCI	0E	Number of bytes per FBA block or PAM page (2048 if TDCFFBA is set).
TDCABPT	10	Number of Associator blocks per track.
TDCABS	12	Associator block size.
TDCACPB	14	Number of FBA blocks or PAM pages per Associator block (if TDCFFBA is set).
TDCDBPT	16	Number of Data Storage blocks per track.
TDCDBS	18	Data Storage block size.
TDCDCPB	1A	Number of FBA blocks or PAM pages per Data Storage block (if TDCFFBA is set).
TDCWBPT	1C	Number of Work blocks per track.
TDCWBS	1E	Work block size.
TDCWCPB	20	Number of FBA blocks or PAM pages per Work block (if TDCFFBA is set).
TDCTSBPT	22	Number of TEMP or SORT blocks per track (if TDCFFBA is set).
TDCTSBS	24	TEMP or SORT block size.
TDCTSCPB	26	Number of FBA blocks or PAM pages per TEMP or SORT block (if TDCFFBA is set).
TDCPBPT	28	Number of PLOG blocks per track.

Label	Offset	Contents
TDCPBS	2A	PLOG block size.
TDCPCPB	2C	Number of FBA blocks or PAM pages per PLOG block (if TDCFFBA is set).
TDCCBPT	2E	Number of CLOG blocks per track.
TDCCBS	30	CLOG block size.
TDCCCPB	32	Number of FBA blocks or PAM pages per CLOG block (if TDCFFBA is set).

In addition, the length of a sequential protection log block may have to be increased. This length is contained in the corresponding PTT entry in CSECT ADAIOI of the load module ADAIOI. The address of the first PTT entry is contained in the fullword at ADAIOI+X‘E4’.

Each PTT entry is X’10’ bytes long and has the structure given below:

Label	Offset	Contents
PTTPN	00	Program number
PTTFT	01	File type
PTTN	02	DD name characters 2 - 8
PTTF	08	Flags: OUT (X’80’) output BSAM (X’40’) BSAM BACK (X’20’) read backwards JCL (X’10’) BLKSIZE/LRECL/RECFM taken from DATADEF statement or label UNDEF (X’04’) undefined record format VAR (X’02’) variable record format
-	09	Reserved
PTTMBS	0A	Maximum block size
-	0C	Reserved

The PTT entry for the sequential protection log can be identified by X‘12F1’ in its first two bytes.

General Rules for Defining Device Block Sizes

The following general rules must be followed when defining Adabas device block sizes:

- All block sizes must be multiples of 4;
- A single block cannot be split between tracks (i.e., the block size must be less than or equal to the track size.

Block Rules for ASSO/DATA

- Associator block size must be greater than one-fourth the size of the largest FDT, and should be large enough to accept definitions in the various administrative blocks (RABN 1 - 30) and in the FCB;
- The block sizes for Associator and Data Storage should be a multiple of 256, less four bytes (for example, 1020) to save Adabas buffer pool space.
- The Associator and Data Storage block sizes must be at least 32 less than the sequential block size.
- Data Storage block size must be greater than: (maximum compressed record length + 10 + padding bytes).

Block Rule for WORK

The Work blocksize must be greater than either (Maximum compressed record length + 100) or (ASSO block size + 100), whichever is greater.

Block Rules for TEMP/SORT

If ADAM direct addressing is used:

size > (maximum compressed record length + ADAM record length + 24);

size > 277 (maximum descriptor length + 24)

- However, TEMP and SORT are generally read and written sequentially; therefore, the larger the TEMP/SORT block size, the better.
- Block sizes for TEMP and SORT must be greater than the block sizes for Data Storage.

Block Rule for PLOG or SIBA

- PLOG or SIBA block size must be greater than or equal to the Work block size.
- It is also recommended that PLOG/SIBA be defined larger than the largest Data Storage block size. This avoids increased I/O caused by splitting Data Storage blocks during online ADASAV operations.

  The block size (BLKSIZE) of a sequential file is determined as follows:

```
     if PTTF(JCL) then BLKSIZE is taken from file assignment statement or la-
bel;
     if PTTMBS > 0 then BLKSIZE = PTTMBS;
     if PTTMBS = 0 then
       if tape then BLKSIZE = 32760;
       else BLKSIZE = TDCMSBS;
     else if BLKSIZE in file assignment statement or label then use it;
          if PTTF(OUT) then
            if QBLKSIZE > 0 then BLKSIZE = QBLKSIZE;
            if tape then BLKSIZE = 32760;
            else BLKSIZE = TDCMSBS;
          else error.
```

  Note:
  QBLKSIZE is an ADARUN parameter.

Using 3480/3490 Tape Cartridge Compression (IDRC)

The use of hardware compression (IDRC) is not recommended for protection log files.

5

INSTALLING THE AOS DEMO VERSION

This chapter tells you how to install the Adabas Online System (AOS) demo version manually. To install this facility on systems that use Software AG's System Maintenance Aid (SMA), refer to the chapter of this manual devoted to installing Adabas in your particular operating environment.

Note:
To install the full version selectable unit Adabas Online System (AOS), see the Adabas Online System Manual (AOSvrs-030IBB).

The AOS demo version requires Natural version 2.2.8 or above.

For information about SMA, see the *System Maintenance Aid Manual* (SMAvrs-030IBB).

Installation Steps

1. Copy AOSASM into the library NATvrs.MOD without renaming

The Adabas module library ADAvrs.MOD contains the module AOSASM.

2. Perform a Natural INPL

The tape containing the AOS demo version contains an INPL-formatted dataset in Natural 2.2. The programs for the AOS demo version facility are stored in library SYSAOS.

3. Load the error messages

The error messages are stored in an ERRN-formatted dataset included on the tape.

Use the Natural utility ERRLODUS to load the messages.

See the *Natural Utilities Manual* for information about the ERRLODUS utility.

4. Execute the AOS demo version

Log on to the application library SYSAOS and entering the command DBMENU.

Installing with Natural Security

Natural Security must be installed before implementing Adabas Online System Security. See the *Adabas Security Manual* for more information. For information about installing Natural Security for use with AOS Security, see the *Natural Security Manual*.

Natural Security version 2.2.8 or above includes the ability to automatically close all open databases when the Natural command mode's LOGON function of the AOS demo version is invoked.

Use the following procedure if Natural Security is installed in your environment.

1. Define at least the library SYSAOS to Natural Security

Software AG recommends you define this library and any others you may define as protected.

2. Specify the start-up program for SYSAOS as DBMENU

Do not specify a start-up program name for the other libraries.

Setting the AOS Demo Version Defaults

Parameters that control the operation of the AOS demo version can be set at installation time by changing the defaults in the Natural program ADVUEX1. The table below lists the parameters and possible values. Default values are underlined:

Parameter	Valid Values / Default	Function
AOS-END-MSG	Yes (Y) / No (N)	Display the AOS demo version end-of-session message?
AOS-LOGO	Yes (Y) / No (N)	Display the AOS demo version logo?
CPEXLIST	No (N): normal list Yes (Y): extended	Display extended checkpoint list?
MAX-AC-IOS	0-999999 (150)	AC read converter block threshold value
NR-EXT	1, 2, 3, 4, 5	“Critical” extent threshold for listing file
STATINTV	1-9999 seconds (60)	Statistics-gathering interval

To change the defaults, you must edit the Natural ADVUEX1 program and make the changes directly within the program listing in the defaults area, which looks as follows:

```
     .
     .
     .
DEFINE DATA PARAMETER USING ADVPUEX1
END-DEFINE
*
* SET THE DEFAULTS
*
AOS-END-MSG = 'Y'  (Display end-of-session message)
AOS-LOGO = 'Y'     (Online System logo display—set to 'N' for no logo display)
CPEXLIST = 'N'     (Checkpoint list control—set to 'Y' for extended checkpoint list)
NR-EXT = 4         (Critical extent threshold—1, 2, 3, 4 or 5)
MAX-AC-IOS = 150   (AC read converter block threshold)
STATINTV = 60      (Statistic-gathering time. range: 1 - 9999)
*
END
```

6

INSTALLING THE RECOVERY AID (ADARAI)

To install the Adabas Recovery Aid, it is necessary to

- allocate the recovery log;
- customize the skeleton job streams for your installation (see the *Adabas Operations Manual* for more detailed information);
- update the necessary nucleus run/utility job control to include the Recovery Aid data definition statements;
- install the Adabas/ADARAI utility configuration; and
- run ADARAI PREPARE and a save operation to begin a logging generation.

Except for customizing the skeleton job stream, the specific installation steps are as follows:

1. Define and format the recovery log files

The DDRLOGR1 and DDRLOGM1 files must be on the same device type.

Use the ADAFRM RLOGFRM function to format the RLOGs.

Use the ADAFRM RLOGFRM MIRROR parameter to format the DDRLOGM1 file.

2. Add data definition statements for the recovery log files

Add DDRLOGR1 and DDRLOGM1 DD statements to the nucleus job stream and to any utilities that update or save the database and thus write to the RLOG files.

Whenever these utilities are executed while ADARAI is active in the database (that is, after the PREPARE function has been executed), the DDRLOGR1 and DDRLOGM1 DD statements must be included.

The following utilities update the database and therefore write to the RLOG:

ADAORD (all STORE and REORDER functions)
ADALOD (all functions)
ADAINV (all functions)
ADARES REGENERATE/BACKOUT database
ADASAV RESTORE (all functions) and RESTPLOG
ADADEF NEWWORK

The following utilities save the database and therefore write to the RLOG:

ADASAV SAVE (all functions)
ADAORD RESTRUCTURE
ADAULD

The following utility functions have an impact on recovery and therefore write to the RLOG:

ADARES PLCOPY/COPY
ADASAV MERGE

Additionally, the Adabas nucleus writes to the RLOG during startup and termination. The nucleus also writes checkpoint information to the RLOG when ADADBS or Adabas Online System functions are processed, ensuring these events are known to ADARAI for recovery processing.

Step 3. Install ADARAI on the database

Execute the ADARAI PREPARE function.

ADARAI PREPARE updates the ASSO GCB to indicate that ADARAI is installed. It also creates a control record on the RLOG files with necessary ADARAI information (number of generations, RLOG size, etc.).

Step 4. Create the first ADARAI generation.

Execute ADASAV SAVE (database) to start the logging of RLOG information.

See the *Adabas Utilities Manual* for more information.

Note:
Once ADARAI is active in the database, protection logging must always be used.

7

ADABAS DUMP FORMATTING TOOL (ADAFDP)

ADAFDP is the address space dump formatting module. During abnormal shutdown of the Adabas nucleus, this module receives control to format and display information that should help you analyze the reason for the error(s).

During a nucleus shutdown, ADAMPM determines the reason. If the reason is abnormal termination, ADAMPM loads the ADAFDP module into the address space prior to the 20 call to the Adabas Router. ADAFDP subsequently receives control to format nucleus information.

If ADAFDP cannot be loaded, a message ADAF03 is written to the console and abnormal shutdown continues.

Information Formatted by ADAFDP

Much of the information formatted by ADAFDP is self-explanatory. However, because the type and amount of information depends on the shutdown situation, a brief discussion of ADAFDP output is presented.

The following is a breakdown of the information formatted by ADAFDP with a brief description of its content:

1 ADAH51 / ADAH52

The message is displayed on the console and written to DD/PRINT at the point where the format begins and terminates.

2 ADAMPM ABEND CODE and PSW

If an ABEND code and program status word (PSW) were saved in ADAMPM by the Adabas STXIT, ADAFDP displays these. In addition, ADAFDP determines the module whose entry point best fits the PSW and calculates the offset within that module. If the ADAMPM abend code and PSW are zero, ADAFDP does not format this information.

3 ADABAS MODULE LOCATIONS

ADAFDP formats and displays the location of each of the Adabas nucleus modules resident in the address space.

4 ADDRESS LOCATIONS FOR USER EXITS

ADAFDP formats and displays the location of any user exit loaded with the Adabas nucleus.

5 ADDRESS LOCATIONS FOR HYPEREXITS

ADAFDP formats and displays the location of any hyperexit loaded with the Adabas nucleus. Hyperexits 10-31 are displayed as A-U, respectively.

6 ADANC0 STANDARD REGISTER SAVE AREA

Registers 0-7/8-F, which are saved in ADANC0. ADAFDP determines if any of these registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that address. If the register is 12 and it points to a user thread, ADAFDP snaps the entire thread.

7 ADANC0 ABEND SAVE REGISTERS

Registers 0-7/8-F, which are saved in ADANC0 as a result of a user abend. ADAFDP determines if any of these saved registers contains an address that points at a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location. If the saved register is 12 and it points to a user thread, ADAFDP snaps the entire thread.

8 ADAMPM SAVE REGISTERS

Registers 0-7/8-F, which were saved in ADAMPM by the Adabas STXIT. These are the same registers displayed with the ADAM99 message. ADAFDP determines if any of these saved registers contains an address that points within a nucleus pool in storage. If yes, ADAFDP indicates which pool and snaps storage at that location.

9 BEGIN / ENDING ADDRESSES OF POOLS / TABLES

ADAFDP determines begin/ending address locations for pools and tables for the Adabas nucleus. These addresses are presented for easy location in the actual dump. The following is a list of pool abbreviations with explanations:

LOG	Log area
OPR	Adabas nucleus operator command processing area
CQ	Address of the command queue, which is formatted later by ADAFDP
ICQ	Internal command queue
TT	Thread table
IA1	Software AG internal area 1
SFT	Session file table
FU	File usage table
FUP	File update table
IOT	I/O table for asynchronous buffer flushing
PL2	PLOG area for asynchronous buffer flushing
PET	Table of posted ETs
TPT	Tpost
TPL	Tplatz
UQP	Unique descriptor pool
UHQ	Upper hold queue
HQ	Hold queue
UUQ	Upper user queue
UQ	User queue
FP	Format pool
FHF	File HILF element
PA	Protection area
TBI	Table of ISNs
TBQ	Table of sequential searches
WK3	Work part 3 space allocation table
IA2	Software AG internal area 2
WK2	Work part 2 space allocation table
VOL	VOLSER table
WIO	Work block I/O area
FST	Free space table work area
UT	User threads
WP	Work pool

AW2	Work block asynchronous I/O area
IOP	I/O pool related to asynchronous buffer flush
IU2	Buffer pool importance header upper 2
IU1	Buffer pool importance header upper 1
BU2	Buffer pool upper header 2
BU1	Buffer pool upper header 1
BH	Address location of the buffer pool header, information from the buffer pool header is formatted later by ADAFDP
BP	Address location of the physical start of the buffer pool

10 ADABAS THREADS

ADAFDP formats the physical threads including threads 0, -1, and -2. The number of lines depends on the value of NT. The thread that was active at the time of the abnormal termination (if any) is marked by a pointer "-->".

11 USER THREADS

For any of the threads -2 to NT that had assigned work to perform, ADAFDP formats and displays information about the status of that thread. The following is a list of the formatted information:

Thread Number	-2 to NT.	
Status	Indicates the current status of the thread. The following statuses are possible:	
	Active	The currently active thread
	In Use	Thread has been assigned work
	Waiting For I/O	Waiting for a block not in buffer pool
	Waiting For RABN	Waiting for a RABN already in use
	Waiting For Work-2 Area Block	Similar to Waiting For I/O
	Waiting Workpool Space	Provides number of bytes in decimal
	Ready To Run	Waiting to be selected for execution
CMD	The Adabas command being executed.	
Response Code	Response code (if any).	

File Number	File number for this command.	
ISN	Internal sequence number for this command.	
Sub. Rsp	Subroutine response code (if any).	
Last RABN for I/O	Last RABN required by command processing, in decimal.	
Type	Last RABN type (A - ASSO, D - DATA).	
CQE Addr	Command queue element address for this command.	
User Jobname	Job name for user who executed this command.	
ITID	Internal Adabas ID for user who executed this command.	
User	User ID for user who executed this command.	
Unique global ID	28-byte ID for user who owns this command.	
Buffer Addresses	for:	control block
		format buffer
		search buffer
		value buffer
		ISN buffer
Buffer Lengths	FL	format buffer length
	RL	record buffer length
	SL	search buffer length
	VL	value buffer length
	IL	ISN buffer length
Snap Thread	The first 144 bytes of the user thread are snapped.	

12 FOLLOWING COMMANDS WERE FOUND IN THE CMD QUEUE

ADAFDP scans the command queue and formats information for any command found in the queue. If there are no commands in the command queue, ADAFDP skips. The following is a definition of the information formatted:

CQE Address	The address location of this CQE.	
F	Command queue flag bytes.	
	First Byte: General Purpose Flag	
	X’80’	User buffers in service partition, region, address space
	X’40’	ET command waiting for 12 call
	X’20’	Waiting for 16 call

	X’10’	16 call required
	X’08’	Attached buffer
	X’04’	Attached buffer required
	X’02’	X-memory lock held (MVS only)
	Second Byte: Selection Flag	
	X’80’	In process
	X’40’	Ready to be selected
	X’20’	Search for UQE done
	X’10’	UQE found
	X’08’	Not selectable during BSS=x’80’ status
	X’04’	Not selectable during ET-SYNC
	X’02’	Waiting for space
	X’01’	Waiting for ISN in HQ
CMD	The command type.	
File Number	The file number for this command.	
Job Name	Job name for the user.	
Addr User UQE	Address of users UQE, if searched for and found.	
Addr User ASCB	Address location of user’s ASCB.	
Addr ECB	Address location of user’s ECB (in user’s address space).	
Addr User UB	Address of users UB (in user’s address space).	
Addr User PAL	Address location of user’s parameter address list.	
CQE ACA	ACA field of CQE.	
CQE RQST	RQST field of CQE.	
Abuf/Pal	Address of the attached buffer/parameter address list (PAL) for CMD.	
Comm Id	28-byte unique user ID for this command.	

13 POOL INTEGRITY CHECK

ADAFDP check the integrity of several pools within the Adabas nucleus address space. If an error is detected within that pool, ADAFDP indicates which pool and what type of error was encountered. In addition, ADAFDP snaps storage at the location where the error was detected.

14 FOLLOWING RABNS / FILES ACTIVE IN BUFFER POOL

ADAFDP scans the buffer pool header for RABNs that were active or being updated. A list of those RABNs is formatted as follows:

RABN Number	The RABN number in decimal.	
Type	Type of block (A - ASSO, D - DATA).	
Flag	BP header element flag byte.	
	AKZ	X’40’ Active indicator
	UKZ	X’20’ Update indicator
	RKZ	X’10’ Read indicator
	XKZ	X’04’ Access is waiting for block
	YKZ	X’02’ Update is waiting for block
	SKZ	X’01’ Write indicator
File	File number that owns this block.	
Address	Address location of block in storage.	

15 ADAIOR REGS FOUND AT OFFSET X’080’

Registers 0-7/8-F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.

16 ADAIOR REGS FOUND AT OFFSET X’0C0’

Registers 0-7/8-F found saved in ADAIOR at this offset. If ADAFDP determines that any of these register values is pointing within an Adabas pool, it snaps storage at that location.

17 ICCB POINTED FROM X’A0’ IN IOR

The ICCB address to which this offset in ADAIOR points.

18 ADAI22 ADAIOR TRACE TABLE

Format of ADAIOR’s trace table; same as that found with the ADAM99 message.

A

APPENDIX A : ADAUTM FOR BS2000

Software AG provides ADAUTM to coordinate Adabas database operations with the SNI BS2000 systems running the Universal Transaction Monitor (UTM).

This appendix describes ADAUTM and its operation.

Common Synchronization Points

To safely operate SNI's Universal Transaction Monitor (UTM) and a database, both systems must be able to run separately and coordinate their restarts. That is the only way to synchronize a database transaction running inside a UTM transaction.

ADAUTM makes it possible to have a common point of synchronization and thereby to set the whole transaction back to the last common synchronization point, if needed.

Both Adabas and UTM define points of synchronization ("sync points"):

- Adabas defines a sync point at the end of the database transaction;
- UTM defines a sync point at the end of a UTM transaction.

In some cases, these sync points are common to both Adabas and UTM. If an emergency restart should be necessary, both of the systems can set the interrupted transactions back to the last common sync point.

A database connection module (DBCON) is used for communication between UTM, its user program, and Adabas. The use of synchronization mode is controlled by a parameter in the UTM utility KDCDEF.

The UTM Transaction Concept

The UTM transaction concept is to be referenced to the database access here only.

Each UTM user program starts a transaction using an INIT call and ends with a PEND nn. When ‘nn’ takes on the value RE or FI, UTM writes a synchronization point followed by an end-of-transaction for the database transaction. Normally, a user program determines the end of a database transaction. In an Adabas environment, the DBCON can take over that task. After an end-of-transaction ET for a database, no more calls to this database are allowed within the current UTM transaction.

Comments and Limitations

DBCON is invoked from UTM only if it occurs on a TP/DB transaction. If the last transaction within a UTM process is a pure TP transaction, the start parameter VG-ENDE=CL cannot be carried out at the end of the UTM process.

A 16-byte prefix of ET data is written at every sync point, in the order of synchronization. This prefix is transparent to the user. This means that when a UTM process contains more than one UTM transaction and the user wants the ET data to be available after the end of the process, the ET data must be written at every ET or with the last ET/CL for this DB transaction. Otherwise, the former ET data will be overwritten by the prefix.

A UTM process imposes only a few limitations on ADAUTM at this time. Those limitations refer only to the Adabas commands OP, ET, and BT. When using an OP command, the R option cannot be used if a process contains more than one UTM transaction because the record buffer is not available after the end of a database transaction. In case of ET and BT, the ISN hold option cannot be used.

The length of ET data that can be used is limited to 1984 bytes for users.

Functions of ADAUTM

In the database transaction process, all calls to the database are transferred to DBCON to be checked. The start, update, and end of transaction are the most important parts.

At the start, a transaction is checked to see if the first call is an OP command and if so, whether the additions 1 of the Adabas control block contains a valid value. If the additions 1 field does not contain a valid string, a generic one is created and put in. If the first command is not an OP command, such a command is carried out implicitly; after that, the user's order is handed over to Adabas.

All commands that have, either explicitly or implicitly, an attribute update are checked. The database ID of the first commands of that category of a transaction is marked as a update database for this transaction. That means that it is not allowed to carry out updates for more than one database within one database transaction.

Usually, the user initiates the end of transaction by issuing an ET/CL command. If that is not the case, the DBCON generates the UTM end of transaction. At this point, calls to the database are not permitted until the end of the UTM transaction (otherwise, the whole transaction must be set back to the last sync point).

When processing a UTM transaction, the main task is to handle the end of transaction procedure. Here, the synchronization with the database is done. In any case, ET data written for synchronization purposes is transparent to the user. Another listed function is the backout of a TP/DB transaction, as well as the functions with special positions, such as those to process status information. That special function is used during emergency restart before UTM is active.

Establishing and Disestablishing a Connection

Connecting to Adabas

The initial link may be subdivided into the following parts:

- Setting default values that are not defined at session startup, using the appropriate parameters.

  All parameters that did not acquire a value during the start of the UTM session are now initialized by a default value. These defaults are described in a later section.

- Creating the internal administration tables.

  A table is built that contains information about the user and the user's transactions. The table is installed as a common memory pool. The value of the parameter APPLI-ID is used as part of the pool's name.

- Making the database and the running mode available.

Disconnecting from Adabas

When disconnecting from Adabas, the administration pool is closed in accordance with BS2000 conventions.

User Call

All user calls to Adabas pass through this function. Here, all the necessary checks are carried out. These checks refer to the synchronization of the TP and DB transaction, but not to the syntax of the Adabas commands or their buffers. The functionality is described with an example of a UTM transaction that contains accesses to a database.

At the beginning, DBCON determines whether the current database call is the first within the UTM transaction. If it is the first call, DBCON determines whether it is an OP (open) command call to initiate an Adabas transaction. If it is not an OP command, an internal OP is performed after the user's call is completed.

DBCON also determines whether a command leads or could lead to an update in any form. If so, the database ID is recorded. With the occurrence of the first update-type command, DBCON marks the transaction as an update transaction. From that point, updates may only be done on this database during this transaction. Modifications to any other databases are rejected as errors, and the whole transaction is set back.

At the end of a database transaction, the ET/CL (end-of-transaction/close) commands are not passed immediately to Adabas, but instead are delayed until the end of the UTM transaction. This means that between the end of a database transaction and the end of the related UTM transaction, no more calls to the database can be carried out. Any attempt to execute such calls leads to a backout of that transaction. At the end of the UTM transaction, it is determined whether the user has determined the end of the transaction, or if DBCON must end it internally.

User Open

If the field ACBADD1 contains a value equal to **zero** or **blank**, DBCON tries to find an ET data ID in the administration pool (subsequent transaction within a UTM process). When available, this ID is taken over into ACBADD1; otherwise, an ID is built by the constant ADAU, the APPLI-ID, and an internal number. If a user exit is available, it gets control in order to modify the passed Adabas control block. The only change the exit can make is to modify the value in ACBADD1. After the DBCON gets back control, it overwrites the field with the old value if the exit has changed it to zero or blank, or the ET data ID was changed after the first transaction of this process.

Internal Open

If the first database call is not an OP command and ET-MODE=AUTO (the default) is activated, DBCON determines whether or not this is a subsequent transaction within the current UTM process. If it is, the call will be executed. If Adabas returns the call with a response code 9, an internal OP command is created and executed; after that, the user's call is repeated. At the beginning of a UTM process, an internal OP command is created, carried out, and after that, the user call. The internally-generated OP command is passed to the user exit in both cases. See the section **The Call User Exit** on page 64.

UTM End of Transaction

The function PEND with one of the attributes RE, FI, or FC determines the end of transaction of UTM. The TP transaction end also contains an end of a DB transaction. That means that now the commands ET/CL are carried out. These were delivered either by the user, or must be created by the DBCON now. ET/CL commands always get an "E" option in the Adabas control block. Before Adabas is called, the DBCON passes the command through to the exit. After a successful execution of the command with a PEND-RE, data about this transaction is stored in the administration pool.

Backout of a UTM Transaction

This function is called by UTM if a user executes a RESET or PEND-ER call. The DBCON calls this for itself if the database is not available or an error occurs during the end of transaction procedure. Adabas return codes that occur during a transaction are passed directly back to the user and do not lead to a backout of the transaction.

Coordinated Restart

The coordinated restart contains two functions which are called during the phase of emergency restart and/or the normal end of the application. An emergency restart indicates an abnormal end of the last UTM session. At a new start of the application, one of two conditions may occur:

- There are open transactions
- There are no open transactions

While the UTM transaction is still open, it must be determined whether or not the database transaction is also still open. This occurs as follows:

In the case of emergency restart, the DBCON is called by the order Check DB Status for update transaction. At first, the ET data belonging to the update database are read. Then all read-only database operations are handled in the same way. The following situations may occur:

- If the synchronization data of UTM are equal to that read from the update database ET data header, the transaction is marked as finished for UTM.
- If the synchronization data of UTM are not equal, the complete DB transaction is backed out (BT) and UTM is informed with “transaction cancelled”. If Adabas returns a response code unequal to 9 or 22, the process is stopped. The UTM utility KDCDEF may be required.

If there are no open transactions, UTM requires the DBCON to delete all information needed for synchronization. In addition, after a normal end of the last UTM session, the DBCON is required to delete all information.

The Call User Exit (EXIT1)

User exit1 can be used to influence the Adabas commands OP, ET, and CL.

However, user exit 1 can only modify the commands which are offered; it cannot execute its own Adabas command because control is not passed back after executing such a command.

The Adabas control block and the record buffer may be modified depending on the commands being used.

The field E1PARB contains the address of the ET data area provided by ADAUTM. The address at E1PARB points to a field which contains the record length and is a part of a 16-byte header. With the exception of the length field, the header cannot be modified. The current length of the new data, plus 16 (the length of the header) must be set in the length field.

Processing at the Start of a UTM Process

The “read ET data” option with the OP command cannot be inserted by the exit. The value that can be modified is the ET data ID in the additions 1 field of the Adabas control block.

The ET data ID can only be modified at the beginning of a UTM process. It cannot be modified after a PEND RE if the DBCON has to issue an OP (after an Adabas error 9).

When using an OP command, only the fields E1PFB and E1PARB can be modified.

The ET data ID in the additions 1 field can only be modified as follows:

- If bit E1PAR is on, the ET data ID cannot be modified.
- Bit E1POP and E1PIO allow the ET data ID to be modified; however, they cannot be changed to either zero or blank.

Processing at the End of a UTM Update Transaction/Process

An update transaction at the end of a UTM transaction or process can exchange ET/CL commands with each other and/or make ET data available.

At the end of a UTM transaction/process, the following conventions apply:

- User exit1 gets control only for an update transaction.
- The contents of the Adabas command field may only be modified by the exit according to the start parameter VG-ENDE.
- Checks must not occur after returning from the exit.
- The ET data ID may **not** be modified because its range of validity is the UTM process. ET data itself may be changed; the exit must provide the new data in its own area.
- The current length must be delivered in the field Length of the record buffer in the Adabas control block.

If the exit sets the bit E1PFBRE, ADAUTM moves the data into its own area; otherwise, no user ET data are written.

A maximum of 1984 bytes of user ET data are allowed.

If there was an internal ET/CL, the field E1PARB contains low-value.

Parameter List of the User Exit

The following description contains the action fields of the parameter list. The complete list can be taken from the Assembler DSECT EX1PARM.

```
E1PTAS1
E1POP:        User with its own OPEN command.
E1PAR:        If the first command is not an OPEN after PEND RE, it is done
              after response code 9.
E1PIO:        It is provided an internal OPEN command.

E1PFB
E1PFBCO :     EXIT1 has modified the ACB during an OPEN
E1PFBCE :     EXIT1 has modified the ACB during an ET/CL command.
E1PFBRE :     EXIT1 has modified the RB during an ET/CL command.

E1PACB        Address of ACB, either of the user or of ADAUTM.

E1PARB        Address of RB; will be delivered by ADAUTM or is equal to zero
              using an internal ET/CL. The exit 1 provides an address of its
              own record buffer, if an internal ET/CL occurs.
```

The exit is called by the standard BALR interface of the Assembler language. The following registers are used:

R01 : Address of the parameter list
R13 : Address of the save area
R14 : Return address
R15 : Start address of the modules

Installation

To make a coordinated restart possible, the DBCON (ADAUTM) must be specified in the UTM root module of the application. This is done by using the DATABASE statement in the UTM utility KDCDEF. The DATABASE statement defines the database in the source of the UTM root. In addition, the ADAUTM modules and the Adabas link module must be provided for the linkage editor's run.

To install ADAUTM

Step 1 Run KDCDEF with

DATABASE TYPE=DB,ENTRY=xyz

—where "ENTRY=xyz" must match the name used in Adabas calls (usually "ADABAS"). If "ENTRY" is omitted or specified as "DB", "ENTRY=ADABAS" is generated.

The "LIB" parameter of the DATABASE statement does not apply for ADAUTM because ADAUTM must be statically linked with KDCROOT.

Step 2 Assemble KDCROOT with the macro library AUTnnn.MAC where "nnn" is the ADAUTM version, revision, and SM level; for example, "621").

Step 3 Link the UTM application with

- AUTDBmm from AUTnnn.MOD where "mm" is the UTM version (31, 32, 33, 34, and 40 are supported);
- AUTNUC from AUTnnn.MOD; and
- ADALNQ from ADAnnn.MOD.LINKS.

Step 4 Add ADAUTM parameters to theUTM start-up parameters in the UTM start-up procedure.

The ADAUTM Parameters

Syntax

The syntax of an ADAUTM parameter is

```
.DB ADABAS parameter = value
```

If the prefix ".DB ADABAS" is omitted, UTM sends the message K38 -Parameter Error-. The application ends abnormally.

Parameters

Parameter	Use	Possible Value	Default
DATABASE DA DB	Default database ID that is the update database for the current session.	1 - 65536	1

This number is the default database ID during the session.

Parameter	Use	Possible Value	Default
APPLI-ID AID	Required. The short name of the UTM application.	1 - 9999	1

The application ID identifies the UTM application running against Adabas. This value becomes part of the name of the common memory pool. That means that, in accordance with the parameter SCOPE, a default value may lead to an error of the UTM process administration.

Parameter	Use	Possible Values	Default
ET-MODE ETM	ADAUTM generates the OP, ET, CL commands if the application does not do it.	AUTO | MAN	AUTO

Value	Means that . . .
AUTO	an OP command is automatically generated at the beginning of a database transaction, and an ET/CL command is automatically executed at the end of a UTM or database transaction, when necessary. Both are required for synchronization.
MAN	disables this function

Parameter	Use	Possible Values	Default
VG-ENDE VGE	Generate a CL instead of an ET command at the end of the UTM process.	CL | ET	ET

“VGE=CL” generates a CL command for the Adabas transaction if the UTM transaction finishes with PEND FI/FC; otherwise, an ET is generated.

Parameter	Use	Possible Value	Default
UEX1	The name of the user exit that gets control during OP/ET/CL commands.	module name	(none)

Name of the module that is linked from a library with the link name “DDLIB”.

Parameter	Use	Possible Values	Default
SCOPE	Accessibility of the common memory pool for ADAUTM.	USERID | SYSTEM | TASK | USER_GROUP	USERID

SCOPE applies only for TASKTYPE “BATCHTASK” and “TP”. If TASKTYPE is “INTERACTIV”, the scope of the common memory pool is always “TASK”.

Parameter	Use	Possible Values	Default
ESQ	Specify ESQ=NO if ESQ is not installed.	YES | NO	NO

This parameter is used to indicate whether Adabas SQL Server is installed.

Parameter	Use	Possible Values	Default
UID-ADA	How the last 8 bytes of the Adabas communication ID are built by ADAUTM.	KCBENID | KCLOGTER | VGNR	VGNR

Value	Means that . . .
KCBENID	the KB field KCBENID is used
KCLOGTER	the KB field KCLOGTER is used
VGNR	the rightmost 4-bytes are built using the UTM conversation ID, as in releases of Adabas prior to 6.1, and the leftmost 4 bytes (the prefix) are specified by the UID-PRF parameter

Parameter	Use	Possible Values	Default
UID-PRF	If the UID-ADA parameter value is “VGNR”, this parameter specifies the leftmost 4 bytes.	abcd	(none)

Example

```
.DB ADABAS   DB = 002 , AID = 80
.DB ADABAS   VG-ENDE = CL , SCOPE = USERID
.DB ADABAS   ET-MODE = AUTO
.DB ADABAS   UEX1 = ADAEX1
```

Diagnostic Information

ADAUTM diagnostic information is written to UTM's DB-DIAGAREA and to SYSOUT.

UTM's DB-DIAGAREA

The general layout and uses of the DB-DIAG-AREA area are described in the appropriate UTM manual. The DB-DIAGAREA is written as follows:

- **primary** DB trace information provides essential information in an ADAUTM response code.
- **secondary** DB trace information includes more detail; see the DSECT DDBTRAC generated by the macro DDBTRAC for the layout of the information.

Messages to SYSOUT

ADAUTM writes messages to SYSOUT in the format

```
AUTxxxx date time OP=yyyy UID=abcdefgh DBID=nnnnn RSP=mmm
```

—where

xxxx	ADAUTM message code
yyyy	UTM primary opcode; see the UTM macro DBCONPAA
abcdefgh	last 8 bytes of the Adabas communication ID
nnnnn	Adabas database ID
mmm	Adabas nucleus response code

ADAUTM Message Codes

ADAUTM message codes are 4 characters long in the following format:

xnnn

—where

x	type identifier. Possible values are		
	D	(database)	ADAUTM received a response code “nnn” from Adabas. With the exception of D148, Adabas response codes starting with “D” are not reported.
	I	(internal)	an error occurred in ADAUTM’s handling of ET data.
	P	(parameter)	an error occurred in ADAUTM’s start-up parameters.
	S	(system)	ADAUTM received an error from a BS2000 executive macro; or, the installation is not correct.
	U	(user)	ADAUTM received unsupported Adabas calls.
nnn	Adabas response code received by ADAUTM; see value “D” above.		

Rsp	Message	
I100	Internal area for ET data exhausted.	
P100	Statement format invalid.	
P101	Unknown parameter.	
P102	Invalid continuation.	
P103	Prefix “.DB ADABAS” not correct.	
P104	Invalid length of statement.	
P105	Invalid value.	
P120	Value not numeric.	
P121	Numeric value out of range.	
S100	ENAMP error	AUTPOOL
S101	REQMP error	AUTPOOL
S102	DISMP error	AUTPOOL
S106	ESQUTM not linked	AUTCONC
S107	TRACE open error	AUTCONC

S108	ENQAR error	AUTPOOL
S109	DEQAR error	AUTPOOL
S110	ADALNQ not linked	AUTMAIN
S111	Unsupported BS2000 version	AUTMAIN
S112	Unsupported HSI version	AUTMAIN
S113	HSITYPE error	AUTMAIN
S114	TABLE error	AUTMAIN

U100	More than four (4) DBIDs used in a single transaction.
U101	Update command issued between ET and end of UTM transaction.
U102	OP command issued, but ET or CL required.
U103	More than one update DBID used in a single transaction.

Handling Adabas Nucleus Response Codes

When ADAUTM receives response code 148 from the Adabas nucleus, a “DBMS down” condition is reported to UTM.

B

APPENDIX B : SUPPLIED TRANSLATION TABLES

Adabas EBCDIC to ASCII and ASCII to EBCDIC

```
cUES2ASC  DS    0F
c*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c         DC    x'000102033F093F7F3F3F3F0B0C0D0E0F'   0.
c         DC    x'101112133F3F083F18193F3F1D3F1F'     1.
c         DC    x'3F3F1C3F3F0A171B3F3F3F3F3F050607'   2.
c         DC    x'3F3F163F3F1E3F043F3F3F3F14153F1A'   3.
c         DC    x'203F3F3F3F3F3F3F3F3F3F2E3C282B3F'   4.
c         DC    x'263F3F3F3F3F3F3F3F3F21242A293B5E'   5.
c         DC    x'2D2F3F3F3F3F3F3F3F3F7C2C255F3E3F'   6.
c         DC    x'3F3F3F3F3F3F3F3F3F603A2340273D22'   7.
c         DC    x'3F6162636465666768693F3F3F3F3F3F'   8.
c         DC    x'3F6A6B6C6D6E6F7071723F3F3F3F3F3F'   9.
c         DC    x'3F7E737475767778797A3F3F3F5B3F3F'   A.
c         DC    x'3F3F3F3F3F3F3F3F3F3F3F5D3F3F3F'     B.
c         DC    x'7B41424344454647484 93F3F3F3F3F3F'   C.
c         DC    x'7D4A4B4C4D4E4F5051523F3F3F3F3F3F'   D.
c         DC    x'5C3F535455565758595A3F3F3F3F3F3F'   E.
c         DC    x'30313233343536373839 3F3F3F3F3F3F'   F.
c*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
          END
cUES2EBC  DS    0F
c*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
c         DC    x'00010203372D2E2F1605250B0C0D0E0F'   0.
c         DC    x'101112133C3D322618193F27221D351F'   1.
c         DC    x'405A7F7B5B6C507D4D5D5C4E6B604B61'   2.
c         DC    x'F0F1F2F3F4F5F6F7F8F97A5E4C7E6E6F'   3.
c         DC    x'7CC1C2C3C4C5C6C7C8C9D1D2D3D4D5D6'   4.
c         DC    x'D7D8D9E2E3E4E5E6E7E8E9ADE0BD5F6D'   5.
c         DC    x'79818283848586878889919293949596'   6.
c         DC    x'979899A2A3A4A5A6A7A8A9C06AD0A107'   7.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   8.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   9.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   A.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   B.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   C.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   D.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   E.
c         DC    x'6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F6F'   F.
c*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
          END
```

Entire Net-Work EBCDIC to ASCII and ASCII to EBCDIC

```
NW2ASC   DS    0F
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         DC    X'00010203040506070809 0A0B0C0D0E0F' 0.
         DC    X'10111213141516171819 1A1B1C1D1E1F' 1.
         DC    X'00000000000000000000 00000000000000' 2.
         DC    X'00000000000000000000 00000000000000' 3.
         DC    X'20000000000000000005 B2E3C282B5D' 4.
         DC    X'26000000000000000000 21242A293B5E' 5.
         DC    X'2D2F00000000000000007 C2C255F3E3F' 6.
         DC    X'00000000000000000603 A2340273D22' 7.
         DC    X'00616263646566676869 000000000000' 8.
         DC    X'006A6B6C6D6E6F707172 000000000000' 9.
         DC    X'007E737475767778797A 00005B000000' A.
         DC    X'00000000000000000000 00005D0000' B.
         DC    X'7B41424344454647484 9000000000000' C.
         DC    X'7D4A4B4C4D4E4F505152 000000000000' D.
         DC    X'5C7E535455565758595A 000000000000' E.
         DC    X'30313233343536373839 7C00000000FF' F.
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
NW2EBC   DS    0F
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         DC    X'00010203040506070809 0A0B0C0D0E0F' 0.
         DC    X'10111213141516171819 1A1B1C1D1E1F' 1.
         DC    X'405A7F7B5B6C507D4D5D 5C4E6B604B61' 2.
         DC    X'F0F1F2F3F4F5F6F7F8F9 7A5E4C7E6E6F' 3.
         DC    X'7CC1C2C3C4C5C6C7C8C9 D1D2D3D4D5D6' 4.
         DC    X'D7D8D9E2E3E4E5E6E7E8 E9ADE0BD5F6D' 5.
         DC    X'79818283848586878889 919293949596' 6.
         DC    X'979899A2A3A4A5A6A7A8 A9C06AD0A100' 7.
         DC    X'00000000000000000000 00000000000000' 8.
         DC    X'00000000000000000000 00000000000000' 9.
         DC    X'00000000000000000000 00000000000000' A.
         DC    X'00000000000000000000 00000000000000' B.
         DC    X'00000000000000000000 00000000000000' C.
         DC    X'00000000000000000000 00000000000000' D.
         DC    X'00000000000000000000 00000000000000' E.
         DC    X'00000000000000000000 00000000000FF' F.
*                .0.1.2.3.4.5.6.7.8.9.A.B.C.D.E.F
         END
```

C

APPENDIX C : GLOSSARY

The following terms are used in this manual to describe Adabas preparation and installation.

Adalink

The teleprocessing-monitor-dependent interface module that connects the application/user to Adabas.

address converter

Adabas stores each database record in a Data Storage block having a relative Adabas block number (RABN). This RABN location is kept in a table called the address converter. The address converters, one for each database file, are stored in the Associator. Address converter entries are in ISN order (that is, the first entry tells the RABN location of data for ISN 1, the 15th entry holds the RABN location of data for ISN 15, and so on).

address space

The storage area assigned to a program task/work unit.

BUB

The “block of unreadable blocks”. It is contained in the primary ASSO RABN 2 and the mirror ASSO RABN 9 for the primary ASSO, DATA, and WORK; in the primary ASSO RABN 9 and mirror ASSO RABN 2 for the mirror ASSO, DATA, and WORK; and the primary and the mirror PLOGn RABN 1 for PLOGn.

communicator

A routine for communicating between operating systems, making remote targets accessible. Entire Net-work is a communicator.

database administrator

Controls and manages the database resources. Tasks include defining database distribution structure and resources, creating and maintaining programming and operation standards, ensuring high performance, resolving user problems, defining and teaching user training, controlling database access and security, and planning for growth and the integration of new database resource applications and system upgrades. Also known as the database analyst.

ID

An abbreviation of “target ID”, a unique identifier used for directing Adabas calls to their targets.

ID table

A reference data list maintained for all active targets within the boundaries of one operating system. The ID table is located in commonly addressable storage.

IIBS

The “isolated ID bit string”, a 256-bit (32-byte) string contained in the ID table header. Each bit corresponds in ascending order to a logical ID. If the bit is “1”, the corresponding ID is isolated.

isolated ID

The ID of an isolated target, which can be specified by the user as a logical ID. An isolated ID must be greater than zero and less than 256. The isolated ID is interpreted as a physical ID for addressing the target.

isolated target

A target called directly by a user.

logical ID

A user’s identifier of target(s) to which a message is directed. It must be greater than zero (0) and less than 256 (either explicitly or implicitly, the content of the first byte of ACBFNR is a logical ID).

MIRTAB

The “mirror table”, which indicates the status of primary RABNs. It is contained in the primary and the mirror ASSO RABN 7 for ASSO, DATA, and WORK; and the primary and the mirror PLOGn RABN 1 for PLOGn.

non-DB target

A target that is not an Adabas nucleus. Access and X-COM are non-DB targets.

physical ID

The identifier of a target. It must be greater than zero (0) and less than 65,536. A database ID (DBID) is a physical ID.

pseudo-cylinder

The logical cylinder on an fixed-block-addressed (FBA) device that has no actual DASD cylinder.

reset

A flag bit is said to be reset when it contains 0.

router

A central routine for communication within the boundaries of one operating system. The routine is called by users with Adalink routines, and by targets with ADAMPM. The router’s main purpose is to transfer information between the Adalink and Adabas. The router also maintains the ID table. The BS2000 router is the ADARER module, which is loaded into common memory defined by the ADARUN/ADALNK parameter IDTNAME.

service

A processor of Adabas calls and issuer of replies. An Adabas nucleus is an example of a service (see also target).

set

A flag bit is said to be set when it contains 1.

target

A receiver of Adabas calls. A target maintains a command queue, and communicates with routers using ADAMPM. A target is also classified as a service (see definition). The Adabas nucleus is a target.

user

A batch or online application program that generates Adabas calls and uses an Adalink for communication.

INDEX

A

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

T

U

W

X

Z